What is claimed is:

1. A system for developing distributed applications over a network of computing units, the system comprising:
    a. a plurality of component programs installed over the network of computing units to create the distributed application;
    b. a plurality of data stores on one or more of the computing units comprising a plurality of routes for data transfer between the component programs and a plurality of parameters for configuring the component programs; and
    c. a plurality of controller programs running on one or more of the computing units in the network for interacting with the component programs and for interacting with other controller programs to send and receive data by referring to routing information from the data stores.

2. The system as recited in claim 1, wherein the component programs perform pre-determined computational logic on data streams presented on their input ports.

3. The system as recited in claim 1, wherein the component programs do not have any memory data pertaining to the routing of data between component programs in the distributed application.

4. The system as recited in claim 1, wherein the component programs do not have any memory data pertaining to locations of any other component programs in the distributed application.

5. The system as recited in claim 1, wherein the component programs do not have any memory data pertaining to inter-relationships and the organization of other component programs in the distributed application.

6. The system as recited in claim 1 wherein the component programs are adaptors for communicating with external applications that are not installed within the system.

7. The system as recited in claim 1, wherein the data store containing the message routes between the component programs is in extensible mark up language.

8. The system as recited in claim 1, wherein the data store can be changed by adding, removing or hot-swapping component programs and message routes while the distributed application is running.

9. The system as recited in claim 1, wherein the data store can be replicated for high availability on a multiplicity of computing units.

10. The system as recited in claim 1, wherein the controller program is a daemon program.

11. The system as recited in claim 1, wherein the controller program of the computing unit on which the distributed application is created disseminates the routing information in the data store to all the other controller programs that interact with the component programs that form part of the distributed application.

12. The system as recited in claim 1, wherein the controller program sends and receives data streams to and from the component programs installed on its computing unit or on other computing units, using an Application Programming Interface.

13. The system as recited in claim 1, wherein the controller program sends and receives data streams to and from the component programs deployed on its computing unit as binary streams.

14. The system as recited in claim 1, wherein the controller program sends and receives data streams to and from the component programs deployed on its computing unit as XML streams.

15. The system as recited in claim 1, wherein the controller program has access to a message queue for sending and receiving data streams to and from other controller programs.

16. The system as recited in claim 1, wherein the component program has access to a message queue for sending and receiving data streams to and from their controller programs.

17. The system as recited in claim 1, wherein the component program accesses the message queue with network publish and network retrieve commands without addressing or routing parameters.

18. The system as recited in claim 1, wherein the controller program sends and receives data from other controller programs through a message bus.

19. The system as recited in claim 1, wherein the controller program sends and receives data to and from other controller programs directly.

20. The system as recited in claim 1, wherein the controller program launches the component programs that have been installed on its computing unit in the network by giving a launch command.

21. The system as recited in claim 1, wherein the controller program launches the component programs that have been installed on other computing units in the network by forwarding a launch command to the other controller programs running on those computing units.

22. The system as recited in claim 1, wherein the controller program monitors the run-time status of the component programs and the message routes between

them and sends this information to other controller programs through the message bus.

23. The system as recited in claim 1, wherein the controller program monitors and controls component programs by gathering the run time data of all the computer programs in the distributed application.

24. The system as recited in claim 1 wherein the controller program controls networking hardware, storage hardware, instruments and machines connected to the computing units.

25. The system as recited in claim 1, wherein the controller program can be used to control the component programs running on that computing unit as well component programs running on other computing units in the network.

26. A method for developing distributed applications over a network of computing units, with one or more computing units having a controller program running on it, the method comprising steps of:

a. customizing component programs;
b. registering the component programs;
c. composing the distributed application in a data store;
d. checking the connectivity and resources and
e. executing the distributed application.

27. The method as claimed in claim 26, wherein the customizing comprises adding codes to the component programs for building their basic functionality for interacting with the controller programs.

28. The method as claimed in claim 26, wherein the registering comprises steps of:

a. installing component programs on the computing units;
b. specifying the external resources required by the component programs;

c. specifying the input and output channels of the component programs; and

d. making the component programs accessible to one or more the participating computing units in the network.

29. The method as claimed in claim 26, wherein the composing the distributed application comprises steps of:

a. choosing the component programs from set of available component programs;

b. adding and specifying the routes between the component programs;

c. specifying the computing unit on which the component program is to be run;

d. defining run time attributes of the component programs;

e. defining various attributes of the routes; and

f. storing the composed distributed application in a data store.

30. The method as claimed in claim 29, wherein the defining attributes of the routes comprises defining the route type as hub spoke.

31. The method as claimed in claim 29 wherein the defining attributes of the routes comprises defining the route type as peer-to-peer.

32. The method as claimed in claim 26, wherein the composing of the distributed application is done using a GUI.

33. The method as claimed in claim 26, wherein the checking the connectivity and resources comprises steps of:

a. checking if all controller programs are already running on the computing units in the network;

b. checking if all the component programs are installed on the computing units on which they are specified to launch;

c. installing the component programs on the computing units in case they are not already installed;

d. launching of the component programs on the specified computing units; and

e. adding routing information to the internal routing tables of all the controller programs.

34. The method as claimed in claim 26, wherein the executing comprises steps of:

a. receiving of data for the component program by controller programs;

b. collecting of data by the appropriate component program from the controller program

c. processing of data by the component program;

d. receiving of processed data from the component program by the controller program; and

e. passing of the data to the next component program or a plurality of next controller programs based on the routing information stored in the routing table by the controller program interacting with it.

35. The method as recited in claim 34, wherein the receiving comprises placing the data in a message queue corresponding to the component program that is specified to receive this data.

36. The method as recited in claim 34, wherein the collecting of data by the component program comprises continuously polling of the message queue implemented within the controller program corresponding to it, to receive data for performing processing task.

37. The method as recited in claim 34, wherein the receiving of processed data from the component program by the controller program comprises steps of:

a. writing of processed data by the component program to a socket connection established with the controller program; and

b. tagging the processed data received by the controller program with the details of source and output channel of the component program.

38. The method as recited in claim 34, wherein the passing of the processed data to the next component program based on the routing information stored in the routing table by the controller program comprises steps of:

    a. tagging the processed data with the name of the destination component program;

    b. placing the processed data by the controller program in the message bus for the controller program on whose node the destination component is installed for fetching, where the routing table specifies a hub/spoke route type; and

    c. sending the processed data by the controller program directly to the controller program on whose computing unit the destination component program is installed, where the routing table specifies a peer to peer route type.

39. A computer program product for developing distributed application over a network of computing units, the computer program product embodied on one or more computer readable media and comprising:

    a. a computer readable program code means for customizing component programs;

    b. a computer readable program code means for registering the component programs;

    c. a computer readable program code means for composing the distributed applications in a data store;

    d. a computer readable program code means for checking the connectivity and resources of the distributed application; and

    e. a computer readable program code means for executing the distributed application.

40. The computer program product as recited in claim 39, wherein the computer readable program code means for customizing comprises computer readable program code means for adding codes to the component programs for building their basic functionality for interacting with the controller programs.

41. The computer program product as recited in claim 39, wherein the computer readable program code means for registering comprises:

a. a computer readable program code means for installing component programs on the computing units

b. a computer readable program code means for specifying the external resources required by the component programs;

c. a computer readable program code means for specifying the input and output channels of the component programs; and

d. a computer readable program code means for making the component programs accessible to all the participating computing units in the network.

42. The computer program product as recited in claim 39, wherein the computer readable program code means for composing the distributed application comprises:

a. a computer readable program code means for choosing the component programs from a set of available component programs;

b. a computer readable program code means for adding and specifying the routes between the component programs;

c. a computer readable program code means for specifying the computing unit on which the component program is to be run;

d. a computer readable program code means for defining run time attributes of the component program;

e. a computer readable program code means for defining various attributes of the routes; and

f. a computer readable program code means for storing the composed distributed application in a data store.

43. The computer program product as recited in claim 39, wherein the computer readable program code means for checking connectivity and resources comprises:

a. a computer readable program code means for checking if all controller programs are already running on the computing units in the network;
b. a computer readable program code means for checking if all the component programs are installed on the computing units on which they are specified to launch;
c. a computer readable program code mans for installing the component programs in case they are not already installed;
d. a computer readable program code means for launching of the component programs on the specified computing units; and
e. a computer readable program code means for adding routing information to the internal routing tables of all the controller programs.

44. The computer program product as recited in claim 39, wherein the computer readable program code means for executing comprises:
a. a computer readable program code for receiving of data for the component program by controller programs;
b. a computer readable program code for collecting of data by the appropriate component program from the controller program
c. a computer readable program code for processing of data by the component program;
d. a computer readable program code for receiving of processed data from the component program by the controller program; and
e. a computer readable program code for passing of the data to the next component program based on the routing information stored in the routing table by the controller program interacting with it.

45. The computer program product as recited in claim 44, wherein the computer readable program code means for receiving comprises a computer readable program code means for placing the data in a message queue corresponding to the component program that is specified to receive this data.

46. The computer program product as recited in claim 44, wherein the computer readable program code means for collecting of data by the component program comprises a computer readable program code means for continuously polling of the message queue implemented within the controller program corresponding to it, to receive data for performing processing task.

47 The computer program product as recited in claim 44, wherein the computer readable program code means for receiving of processed data by the controller program from the component program comprises:

a. a computer readable program code for writing of processed data by the component program to a socket connection established with the controller program; and

b. a computer readable program code for tagging the processed data received by the controller program with the details of source and output channel of the component program.

48. The computer program product as recited in claim 44, wherein the computer readable program code means for passing of the processed data to the next component program or to a plurality of component programs based on the routing information stored in the routing table by the controller program comprises:

a. a computer readable program code for tagging the processed data with the name of the destination component program;

b. a computer readable program code for placing the processed data by the controller program in the message bus for the controller program on whose node the destination component is installed for fetching, where the routing table specifies a hub/spoke route type; and

c. a computer readable program code for sending the processed data by the controller program directly to the controller program on whose computing unit the destination component program is installed, where the routing table specifies a peer to peer route type.